UNCLASSIFIED

AD-A285 421

AR-008-867

DSTO

Information Technology Division

DTIC ELECTE OCT 1 3 1994 S G D

ELECTRONICS RESEARCH LABORATORY

GENERAL DOCUMENT
ERL-0826-GD

iMAPS GENERAL INTRODUCTION

by

Martin Burke

94-32059

APPROVED FOR PUBLIC RELEASE

UNCLASSIFIED

94

DTIC QUALITY INSPECTED 3

UNCLASSIFIED

AR-008-867

DSTO
AUSTRALIA

ELECTRONICS RESEARCH LABORATORY

Information Technology Division

Accesion For	
NTIS CRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

GENERAL DOCUMENT
ERL-0826-GD

iMAPS
GENERAL INTRODUCTION

by
Martin Burke

SUMMARY

This paper introduces the main ideas of the iMAPS Task which aims to establish an integrated approach to the description, measurement, assessment and prediction of the attributes of software.

AUG 94

APPROVED FOR PUBLIC RELEASE

POSTAL ADDRESS: Director, Electronics Research Laboratory, PO Box 1500, Salisbury, South Australia, 5108.

Contents

Page No.

FIGURES

Executive Summary

Software development is concerned with the synthesis of products to satisfy the needs of customers and users. The development process involves the production and checking of many intermediate products and consumes (or uses) various resources.

Controlled software development involves the repeated assessment of how well the development process has progressed and the prediction of whether the final product will ultimately satisfy its needs and give value for money.

Assessment requires the collection of appropriate evidence. Evidence is usually in the form of either descriptions or measurements. Prediction involves making forecasts. Forecasts are usually in the form of descriptions or quantified statements.

The iMAPS approach sets out to provide an <u>integrated</u> means of describing, measuring, assessing and predicting the attributes of software which deals with the <u>uncertainties</u> involved in a rigorous manner. If successful, the approach will optimise what is described and measured to make increasingly dependable assessments and predictions.

iMAPS promises gains in value for money through enhanced control. If successful it will establish DSTO at the forefront of a field of fundamental importance to the discipline of Software Engineering.

The initial foci of the work will be the software attributes:

- reliability;
- cost.

THIS IS A BLANK PAGE

1. Introduction

1.1 iMAPS

The iMAPS Task, [16], is concerned with devising an integrated approach to the description, measurement, assessment and prediction of the attributes of software. (iMAPS is an acronym for integrated Measurement, Assessment and Prediction of Software.)

This is the first of a series of reports and papers to be produced under the iMAPS task. It provides a general introduction to the background, aims, basic concepts and strategies of iMAPS.

This paper has six main sections. Section 1 provides introductory information on the iMAPS Task's background, purpose and major ideas. Section 2 outlines the approach that will be adopted in undertaking the work involved in the Task. Section 3 gives more detailed information on the methods which will be used. Section 4 lists the tangible products which the Task can be expected to develop. Section 5 briefly discusses the possibilities for collaborative work which are open to the iMAPS Task. Section 6 summarises opportunities for commercial exploitation represented by iMAPS. The paper concludes with a detailed list of the references made in the body of the paper.

This paper has been prepared to be read by:

(a) ITD/ERL management;

(b) members of the iMAPS team;

(c) potential iMAPS Task sponsors;

(d) potential collaborators.

No specific background knowledge is assumed of the reader.

If subsets of this readership require it, specialised variants of the paper will be prepared to emphasise or elaborate aspects of the work introduced in this initial, general variant.

1.2 Task history

The iMAPS Task is to be conducted by Software Engineering Group, Information Technology Division. This section gives a brief history of the iMAPS Task which stresses its relationships with other Tasks conducted by the Group .

The early tasks DST 88/017 and DST 89/047 (both titled "Software Engineering Research") established an initial software engineering technology base, and also spawned several specialised R&D tasks.

Task ALO 91/534 ("Software Measurement and Analysis Technology, (SMAT)" was set up to address the analysis and evaluation of software quality. Some of its areas of research include:

(a) software description and visualisation;

(b) software measurement and tools;

(c) evaluation techniques and tools;

(d) process and environment support.

The Software Evaluation Environment for Ada (SEE-Ada), [12, 19, 27], is a product of SMAT research.

Task DST 92/042 ("Software Engineering Technology Base") continued the development and maintenance of the SE technology base by conducting studies, investigations and other activities in key areas of software engineering not previously covered by the specialised tasks referred to above. iMAPS was one of these activities which complemented the work done under the SMAT Task.

In July 93 Task DST 93/208 ("Advanced Software Engineering Technology") was set-up to maintain and enhance the SE technology base. In November 93 Task DST 93/949 ("integrated Measurement, Assessment and Prediction of Software"), the iMAPS Task, was set-up as a separate Task to pursue specialised research activities.

It is the intention that sponsorship for the iMAPS Task be sought in the future.

1.3 Motivation

Software is an increasingly important element in modern Defence systems. A large proportion of the Defence Organisation's budget is currently committed to the procurement and maintenance of software intensive systems Examples which highlight this include the Submarine, Frigate, Jindalee and Nulka projects.

Some of the major problems facing the procurement, development, operation and maintenance of software in Defence applications are summarised below.

1.3.1 Description

The dearth of sound methods for the unambiguous, consistent and concise description of software frequently gives rise to costly mistakes being made at all stages of the development of software. Mistakes of this nature would be unacceptable in the more mature engineering disciplines. For example, there is no commonly accepted method for describing important software attributes such as reliability, maintainability, safety and security. As such, these aspects are often neglected when establishing requirements for software systems and therefore cannot be assessed during development and prior to acceptance.

1.3.2 Measurement

The inappropriateness of existing software measurement techniques give rise to difficulties in specifying, planning and controlling software development projects. For example, one of the major difficulties experienced in planning new software development projects is the lack of suitable, objective measurements of the 'size' of previous projects to be used in comparative analyses.

Furthermore, a survey conducted in 1990 by the ESPRIT II Project METKIT into software measurement, [30], which covered the Information Technology departments of 400 European organisations and 445 universities worldwide, found that only 5% of the IT departments surveyed used measurement to improve their software development processes. The main reason given for this being that the organisations simply didn't understand how to use measurement effectively.

1.3.3 Prediction

The inability to make confident predictions of the likely properties of software in advance of its development, particularly dependability and cost, commonly lead to high levels of operational and programme risk being tolerated in projects involving software development. For example, METKIT, [30], found that discrepancies of 200% - 300% between forecast and actual cost of development are commonplace in the European software industry. Another example of the poor quality of software predictions is given by a study performed on behalf of the US Air force, [31]. In a survey of software size estimation tools based on a single example product, it found that only two of the tools tested produced estimates within 30% of the actual value and that estimates produced by other tools gave 100% - 200% discrepancies.

1.3.4 Assessment

The lack of a consistent approach to the objective assessment of the qualities of software, both during its development and after it has been completed and delivered, often leads to loss of control of software intensive projects. For example, it is common for the Defence Organisation in the procurement of software systems to trust software development contractors to adopt 'good industrial practices' in the development processes they adopt. As a result of inadequate monitoring/assessment of these processes it is a frequent occurrence for products to be delivered with do not perform the required functions or which have sub-optimal performance characteristics. Accordingly, high maintenance costs are incurred. This is exemplified by the Nulka Project, see [24].

1.3.5 Integration

Software development is concerned with the synthesis of products to satisfy the needs of customers and users. The development process involves the production and checking of many intermediate products and consumes (or uses) various resources.

Controlled software development involves the repeated assessment of how well the development process has progressed and the prediction of whether the final product will ultimately satisfy its needs and give value for money.

Assessment requires the collection of appropriate evidence. Evidence is usually in the form of either descriptions or measurements. Prediction involves making forecasts. Forecasts are usually in the form of descriptions or quantified statements.

At present there is no integrated approach to software description, measurement, assessment and prediction. Therefore, software projects are unable to optimise their description and measurement activities to achieve the most dependable assessments and predictions. This results in inefficiencies and inaccuracies in the control of the projects.

1.4 Objectives

The prediction of various attributes of software very early in their life-cycles is an extremely desirable but elusive objective in software engineering. Software measurement and analysis tools, for example SEE-Ada, [19], AdaMAT/D, [10], can provide very detailed descriptions but only later in the life cycle when detailed information becomes available. Popular software estimation tools, for example COCOMO, see [20], rely on human input of certain numerical factors, derived from experience with previous similar software projects. What is required is a more systematic process for software measurement, in order to support robust predictions at all stages of software development.

iMAPS aims to reduce the costs and risks associated with acquiring, developing and maintaining software by enabling improved levels of control over these processes. iMAPS is intended to benefit all procurers, producers and users of Defence software irrespective of the software's application or the technologies used in the software's development.

The primary goal of the iMAPS Task is to devise an integrated approach to the description, measurement, assessment and prediction of the attributes of software which provides:

(a) consistent means for the description and measurement of software products and processes;

(b) a framework for the collection, storage and analysis of data from software development and maintenance projects;

(c) a framework for monitoring and assessing software as it is being developed and maintained;

(d) a rigorous means for making predictions of the likely values of the attributes of software based upon measured data regarding the software under investigation and other software projects.

If successful, this integrated approach will optimise what is measured to make increasingly dependable assessments and predictions.

Initially, the research is to focus on the key software attributes of :

(a) reliability;

(b) cost.

Figure 1 depicts the iMAPS approach using a spectrometer analogy.

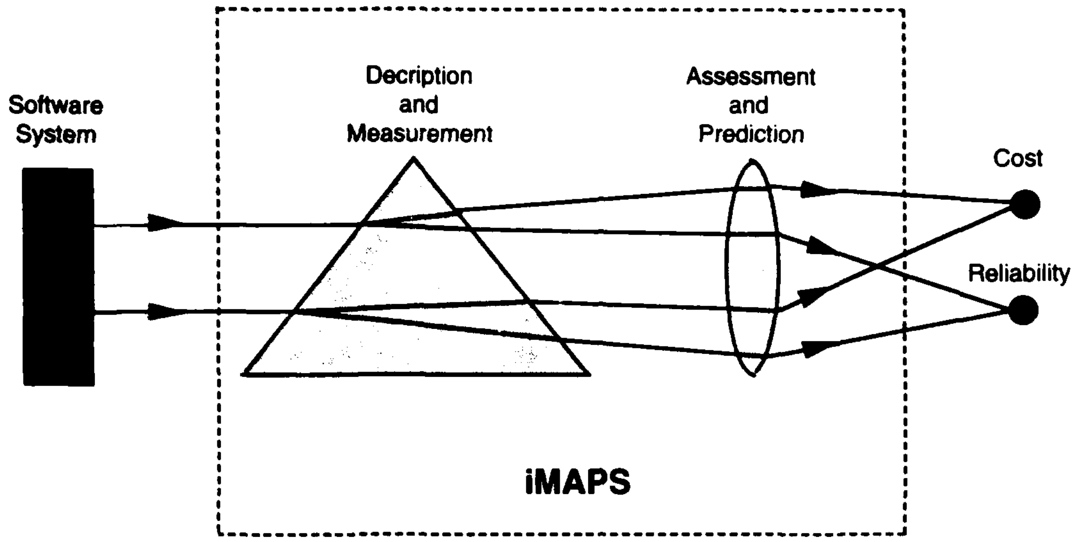

Figure 1 The iMAPS approach

The research will involve:

(a) the development of a conceptual model for iMAPS;

(b) the production/re-use of knowledge-based software tools to support this model;

(c) progressive enhancement of the model and tools on the basis of feedback from a series of experiments conducted on the model through the medium of the tools;

(d) involvement of DSTO and ADF clients.

The objectives of the task are accordingly:

(a) to devise and implement an integrated approach to the description, measurement, assessment and prediction of the attributes of software;

(b) to validate the approach through a Defence oriented case study;

(c) to transfer iMAPS technology to Defence and to industry.

1.5 Use

The scheme is intended to be used at all stages of the software life-cycle including:

(a) feasibility study;

(b) contract negotiation;

(c) requirements analysis;

(d) specification;

(e) design;

(f) coding;

(g) checking;

(h) maintenance;

(i) removal from service.

It is stressed that the iMAPS scheme is an approach rather than merely a software tool. The scheme provides models and methods which are intended to be supported by guides and manuals as well as software.

The scheme is intended to be used by individual software projects but coordinated from a DSTO directed centre. Data is to be collected and stored locally but collated and analysed at the centre. The centre will also:

(a) disseminate and exploit the results of such analysis;

(b) control the development, distribution and support of the iMAPS models, methods and tools;

(c) provide guidance and support in the use of the scheme.

The establishment of such a centre is not in the scope of the current Task, although details of its requirements and methods of working are expected to emerge as a consequence he planned activities. It is not anticipated that it will be appropriate for the centre to be funded, managed or staffed by DSTO. It is more likely that the centre would need to be sponsored separately or run on a commercial basis.

The iMAPS scheme is to be used in:

1.5.1 Description

iMAPS aims to support software description by providing a consistent means of describing what software is required to do, how it does this and how well it does this. This is to be achieved through the provision of:

(a) a consistent terminology and taxonomy for describing software in words. This will emerge as an outcome of the attribute maps, see Section 3.4.1, developed for the iMAPS conceptual model and will be in accord with the work on Software Description being undertaken as part of the SMAT Task, [25];

(b) computer-based methods of integrating and tailoring information about software for specific needs and tasks. One such approach is Perspective-Oriented Description, [12] which is being researched by Rudi Vernik;

(c) examples of how to use iMAPS in the description of software.

1.5.2 Measurement

Software measurement theory, [26, 17], is central to the iMAPS concept. Used wisely it can be used to provide concise objective descriptions which, in conjunction with appropriate models, can provide more accurate and testable predictions than would otherwise be possible. Before discussing these issues further it is necessary to introduce a few basic concepts.

Fenton, [17], makes the following definitions.

Direct measurement *of an attribute is measurement which does not depend on the measurement of any other attribute.*
Indirect measurement *of an attribute is measurement which involves the measurement of one or more other attributes.*

Both direct and indirect measurement of an attribute can be possible, although not necessarily at the same time. Furthermore, more than one form of both direct and indirect measurement can be possible for an attribute, although again it may not be possible to make these measurements at the same time. Direct measurements tend to be more accurate and precise than indirect measurements, although this is not necessarily the case.

There are many circumstances in the iMAPS domain in which it is not possible to make a direct measurement of an attribute of interest. Indirect measurement must suffice until direct measurement becomes feasible.

iMAPS aims to give direction regarding what, how and when software properties should be measured in order to make robust predictions of other, usually more abstract, software properties which cannot be measured directly at that time.

iMAPS sets out to define a base set of software measures. The set will comprise both direct and indirect measures. The definitions will cover:

(a) individual measures;

(b) how to make the measurements;

(c) the relationships between measures.

A considerable amount of work has been conducted in this respect already by SEI, [34] IEEE, [35-37], CECOM, [38], AMI, [40], etc. iMAPS will make as much use as is possible of this existing knowledge making appropriate refinements as necessary.

iMAPS will also attempt to provide:

(a) guidance on how this set can be 'tailored' for the purposes of individual projects;

(b) software tools which assist or automate the measurement process and facilitate subsequent data collation and analysis;

(c) guidance on the use of these tools;

(d) examples of how to use the iMAPS scheme in software measurement.

It will provide guidance on how subsets of the base set of measures can be chosen which, for a prediction of the value of a specified attribute of interest, give:

(a) maximum influence on the uncertainty in prediction;

(b) minimum uncertainty in prediction;

(c) maximum predictive power, where the term 'predictive power' is informally defined as the quotient of uncertainty in a prediction to the 'cost' of the direct measurements used in making it.

1.5.3 Prediction

iMAPS aims to support predictions regarding software by providing forecasts of the (likely) values of software properties which cannot be measured directly, by using data regarding the values of software properties which have been measured directly.

iMAPS predictions are inferences drawn from general and comparative knowledge in conjunction with specific evidence regarding the item of software being analysed. In iMAPS this knowledge is encapsulated in rule maps and the gathering of data is controlled with evidence collection schemata.

The forecasts take into account evidence of how the item of software under investigation has been made and checked and knowledge of how other similar software behaved and/or what properties this software had and how it was made, checked and used.

iMAPS predictions will have the following properties:

(a) quantitative;

(b) qualified by a measure of uncertainty which reflects the imperfection of the knowledge and evidence used in making the predictions;

(c) auditable;

(d) repeatable.

The Task will provide examples of how to use the iMAPS scheme in software prediction.

1.5.4 Assessment

iMAPS aims to support assessment of software by enabling measurement and prediction of the values of the properties of the software under investigation and comparison of these with the intended, required or desired values of the properties of that software.

iMAPS assessments will be made by collecting evidence regarding the software under investigation using the iMAPS scheme and inferring the values of other software attributes which could not be measured directly at that time using the iMAPS prediction methods. The results of these measurements and predictions will then be compared to their target values and assessments made.

iMAPS will enable:

(a) monitoring and recording of the changing status of the software under investigation throughout the various stages of its development and maintenance;

(b) evaluation of the consequences of changes made to the software;

(c) direction of effort in making changes to achieve required results;

(d) use of objective and subjective evidence;

(e) evaluation of the uncertainty in knowledge and evidence used in making assessments.

This will support assessments of:

(a) software quality;

(b) conformance to standards.

iMAPS assessments will be:

(a) based on testable, measurable statements of requirement;

(b) comparative;

(c) investigative or explorative, ie enable both goal-directed and unfocussed analyses;

(d) auditable;

(e) repeatable.

iMAPS will also attempt to provide:

(a) guidance on the use of software tools to assist in software assessment;

(b) examples of how to use the iMAPS scheme in software assessment.

1.6 Users

The intended users of the iMAPS scheme are:

(a) software procurement staff;

(b) software project management staff;

(c) software development, V&V and maintenance staff;

(d) software quality management staff;

(e) independent software assessors, eg IV&V teams, safety assessment staff, software standards accreditation staff etc.

1.7 Benefits

1.7.1 Defence

iMAPS will help reduce the costs and risks associated with commissioning, developing and maintaining software, by enabling improved levels of control over these processes. iMAPS has the potential to benefit all procurers, producers and users of Defence software irrespective of the software's application or the technologies used in the software's development.

1.7.2 DSTO

iMAPS will establish DSTO at the forefront of a field of fundamental importance to the discipline of Software Engineering.

2. Approach

This Section describes the approach that is to be adopted in first breaking down, and then performing, the work involved in iMAPS.

The work involved in the iMAPS Task is to be treated as a matrix of complementary activities as shown in Figure 2. The orthogonal dimensions of the matrix are the so-called *themes* and *foci* of iMAPS. In addition to these there will be some general activities which are concerned with the work as a whole.

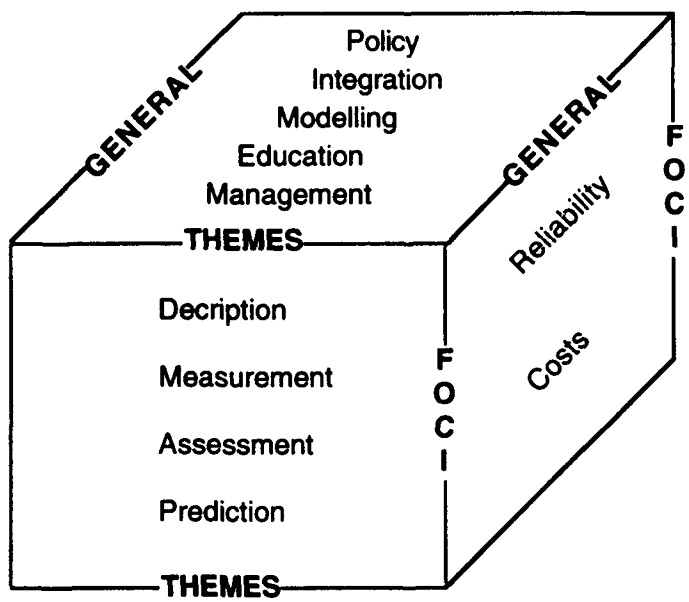

Figure 2 iMAPS Work Breakdown

The iMAPS themes are:

(a) description;

(b) measurement;

(c) prediction;

(d) assessment.

The initial iMAPS foci are:

(a) reliability;

(b) costs.

Breaking the work down in this manner has the following advantages:

(a) it allows the programme to be conducted in a piecemeal manner;

(b) it enables flexible scheduling;

(c) it supports staffing with people with diverse backgrounds and skills.

2.1 General

2.1.1 Modelling

2.1.1.1 Uncertainty

It is essential to the success of iMAPS that a good understanding is formed of the nature of the uncertainty inherent to the knowledge and data that it is to use. Furthermore, it will be necessary to decide which is the most appropriate approach to be adopted in treating such uncertainty in the development and use of the iMAPS scheme. Earlier work on software dependability modelling, [13], and cost estimation, [32, 33], are anticipated to be of direct relevance in this respect.

2.1.1.2 Knowledge acquisition

It will be necessary to study and identify appropriate means of knowledge acquisition for use in modelling. It is not expected that any formal techniques will be used in the preliminary stages other than structured interviews/modelling sessions. A survey will be conducted in an attempt to identify more formal techniques which can be used in later stages. Any processes or techniques developed or used in the course of the Task will be reported as appropriate.

2.1.1.3 Knowledge representation

It will be necessary to study and identify appropriate means of knowledge representation for use in modelling.

The principal contender is the Conceptual Modelling technique developed by Burke in [13]. In this a conceptual model is taken to have three components, namely:

(a) an attribute map;

(b) a rule map;

(c) evidence collection schemata.

The significance of each of these is explained in Section 3.4.

Details of the knowledge representation techniques that they use are given in [13]. These techniques are expected to be enhanced during the course of the iMAPS work and will be reported as appropriate.

2.1.1.4 Reasoning

It will be necessary to study and identify appropriate means of reasoning with uncertain knowledge and evidence for use in the conceptual model.

The principal contender is the inference mechanism of Support Logic Programming, [1-14].

2.1.2 Integrated MAPS

The iMAPS conceptual model will be composed of a small number of integrated, sub-models. Each sub-model will address a sub-domain within the iMAPS scope. The scope of each sub-domain will be a particular high-level software attribute.

The sub-domains which will be modelled in the first instance are those of software reliability and software costs.

2.1.3 Policy shaping

In the assumption that the ideas underpinning iMAPS can be shown to be sound, the success of the iMAPS scheme within Defence will depend, to a large degree, on the extent and alacrity with which the scheme is accepted and utilised by projects and their contractors. The best way to effect this is by the introduction of Department-wide requirements in this respect. Accordingly, efforts will be made to encourage appropriate authorities within Defence to make policy statements which require iMAPS methods to be deployed.

2.1.4 Publicity/education

If the iMAPS scheme is to gain acceptance then it is essential that its intended users are made aware of its potential benefits and that they understand its basic ideas. In order to achieve this a programme of publicity and education will be undertaken. The timing and content of the

programme will need to be carefully coordinated with the other aspects of the Task if best results are to be achieved.

2.1.4.1 Internal seminars

A series of internal seminars are planned. Some of these are intended for a general DSTO audience, others will be restricted to members of the iMAPS team. Topics which will be covered include:

(a) concepts of uncertainty;

(b) mathematics of uncertainty;

(c) uncertainty in software;

(d) mathematics of uncertainty in software;

(e) iMAPS concept;

(f) FRIL;

(g) software reliability attribute map;

(h) software reliability rule map.

2.1.4.2 External

Papers will be submitted for consideration for publication in scientific journals and conferences as appropriate.

2.1.5 Management

iMAPS is to be set up initially as a DSTO sponsored Task with the intention of seeking sponsorship from other parts of Defence as soon as possible. When further particular client related activities begin to emerge individual Defence sponsored Tasks will be set up to service these needs.

The work is well-suited to be tackled by a small core team provided that other parts of the Defence Organisation can be actively involved.

The Task is to run as a number of distinct Sub-Tasks which will address well-defined aspects of the iMAPS work. A Sub-Task will address one (or part of one) of the themes or foci discussed in Section 3. The Sub-Tasks will be managed by designated Sub-Task leaders who will report directly to the Task Manager. An individual may be responsible for the management of more than one Sub-Task. This will enable the Task to be tightly coordinated and controlled but retain the

flexibility to adapt to any changes in the objectives or resources which may arise. It is likely that the Task Manager will also manage some Sub-Tasks.

At sometime in the future it may be advantageous to reorganise the current SMAT and iMAPS Tasks, see [25] and [16], in order to focus effort on research issues which emerge which are of particular interest or benefit.

2.2 Themes

2.2.1 Description

Both the SMAT and iMAPS Tasks will be undertaking work on software description. Furthermore, Rudi Vernik of Software Engineering Group is researching software description as the topic of his PhD studies. The three strands of investigation are intended to share a common conceptual basis which will be applied in different ways.

Coordination of the work is assured by virtue of the facts that Rudi Vernik is the SMAT Task manager, Martin Burke is the iMAPS Task manager and Martin is acting as industrial supervisor of Rudi's PhD studies.

2.2.1.1 Fundamentals

The three work areas in software description referred to above will share a common conceptual basis. A report summarising these fundamental ideas is currently being drafted, [14].

2.2.1.2 Perspective Oriented Description

Perspective Oriented Description is a new approach to providing effective software engineering descriptions which uses computers to provide information in a timely, useable and useful way. It makes use of standard base representations to provide familiar and stable frames of reference for the user. Other aspects of interest can be displayed relative to these base representations, thereby allowing the user to tailor descriptions to specific needs and tasks. Preliminary work has already been published, [12], and has been well received. The SEE-Ada tool developed by Software Engineering Group, [28, 29], is based on this approach.

Perspective Oriented Description is a major aspect of Rudi Vernik's PhD work.

2.2.2 Measurement

2.2.2.1 Attributes

In the early stages of the iMAPS Task independent attribute maps will be developed for the foci of software reliability and software costs. Careful consideration will be made oi how best to integrate these into a general iMAPS attribute map. The relative advantages and disadvantages of keeping these entirely separate and merging them to synthesise an aggregate attribute map will be established.

2.2.2.2 Data collection

The measurement theme will concern itself with specifying, setting-up and coordinating the collection and storage of software measurement information to comply with the principles of the iMAPS scheme.

A staged approach will be adopted. The main stages are:

(a) experimentation within SE Group;

(b) prototype scheme with single Defence client;

(c) widespread use by Defence projects and their contractors.

2.2.2.3 Tools

A survey of software tools for the automatic collection of software measurement data will be conducted. Careful consideration will be made of how the iMAPS approach can make use of these.

M-Base, the commercially available version of the Data Collection and Storage System (DCSS) developed by the ESPRIT II Project MERMAID, will be of particular interest in this respect.

2.2.3 Prediction

2.2.3.1 Predictive power

The prediction theme will be concerned with formulating theory and conducting experiments to investigate how the predictive power of the iMAPS models can be tuned and checked. Data collected in the preceding measurement activities will be used. The same, or similar staged approach will be adopted.

2.2.3.2 Tools

Surveys of software tools for the prediction of software reliability and software costs will be conducted. Careful consideration will be made of how the iMAPS approach can make use of these.

The tools which will be of principal interest in this respect are ESTEEM, [23, 23], QUINCE, [32, 33], and PERFIDE.

2.2.4 Assessment

2.2.4.1 Assessment efficiency

The assessment theme will be concerned with formulating theory and conducting experiments to investigate how effective assessment can be performed. A study will be made of the approaches adopted by expert software engineers when performing software assessments. SEE-Ada, [12, 19], will be the main focus of this activity. Principles of systematic assessment will be formulated which are anticipated to cover:

(a) investigative and explorative assessment;

(b) setting of assessment criteria;

(c) assessment against criteria;

(d) hypothesis formulation;

(e) hypothesis testing;

(f) exploiting tools in assessment.

2.2.4.2 Tools

Careful consideration will be made of how the iMAPS approach can make use of existing software assessment tools. SEE-Ada, [12, 19], will be of principal interest in this respect.

2.3 Foci

iMAPS foci are the (high-level) software attributes of principal interest in an iMAPS conceptual sub-model. They are typically impossible or inconvenient to measure directly and are therefore usually measured indirectly using an inference process and data derived from indirect measurements to which they are related.

Although it is feasible that the iMAPS approach could be used to address any number of high-level attributes, in practice it will concentrate on only a limited number. The high-level attributes which tend to be of prime interest are dependability and costs. The following sections give brief introductions to these concepts/attributes. Each attribute is defined provisionally and is broken down into sub-attributes where appropriate.

In the early phases of the Task only two foci will be addressed, namely software reliability and software costs.

2.3.1 Dependability

System dependability is defined by Laprie, [18], as the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers. The main sub-attributes of dependability are reliability, availability, maintainabilty, safety and security.

Software dependability is defined by Burke, [13], as the degree of corroboration that software possesses its intended properties. It measures:

(a) how much is known about whether the software possesses its intended properties;

(b) how much is known about whether the software does **not** possess its intended properties;

(c) how much is **not** known about whether the software possesses its intended properties or not.

Assessing the dependability of software is the process of estimating these measures by using all available evidence about how the software was developed, including anomaly data, and its operational characteristics.

Previous work conducted by Martin Burke on Software Dependability Modelling and Assessment is anticipated to be of direct relevance to iMAPS. This work was funded by the European Space Agency and SEMA Group and formed the basis of Martin's PhD thesis.

2.3.1.1 Reliability

Definition Software reliability is the probability of failure free operation of a computer program for a specified period in a given environment.

There are various ways in which software reliability can be interpreted. It can be thought of as:

(a) dependability with respect to the continuity of service;

(b) a measure of continuous correct service delivery;

(c) a measure of the time to failure.

2.3.1.2 Availability

Definition Software availability is the probability that a computer program will operate without failure at a specified instant in a given environment.

There are various ways in which software availability can be interpreted. It can be thought of as:

(a) dependability with respect to readiness for usage;

(b) a measure of correct service delivery with respect to the alternation of correct and incorrect service.

2.3.1.3 Maintainability

Definition Software maintainability is the probability that a failed computer program can be restored to failure free operation within a specified period in a given environment.

There are various ways in which software maintainability can be interpreted. It can be thought of as:

(a) the ease with which maintenance actions can be performed;

(b) a measure of continuous incorrect service;

(c) a measure of the time to restoration from the last failure.

2.3.1.4 Safety

Definition System safety is the expectation that a system does not, under defined conditions, lead to a state in which human life or wellbeing is endangered.

There are various ways in which safety can be interpreted. It can be thought of as:

(a) dependability with respect to the non occurrence of catastrophic faults.

(b) a measure of continuous delivery of either correct service or incorrect service after benign failure.

(c) a measure of time to catastrophic failure.

2.3.1.5 Security

Definition System security is the protection of a system from accidental or malicious access, use, modification, destruction or disclosure.

It can be interpretted as dependability with respect to the prevention of unauthorised access and/or handling of information.

2.3.2 Costs

Definition The value of the resources expended in developing, operating and maintaining software.

The main sub-attributes of costs are money and time, with time being broken down further into elapsed time and staff effort.

Previous work conducted by Martin Burke on Software Cost Estimation for the QUINCE Project, [32, 33], is anticipated to be of direct relevance to iMAPS.

3. Method

This Section gives detailed information on specific ideas and methods to be used in the iMAPS Task which are expected to be unfamiliar to the majority of readers.

3.1 Conceptual models

Following Harris, [11], a conceptual model is defined as any structure of knowledge that has the potential for deriving inferences.

Conceptual models emphasise important features of phenomena, either for purposes of detailed explanation or prediction, or as an aid in developing an understanding of those phenomena. A conceptual model of an issue is never an absolutely true account of that issue. When using a model in any of these ways it is important to be aware of the disparity between the model and the reality it attempts to represent.

Key aspects of the conceptual models developed by the author in this and previous work are:

(a) an initial definition of the basic concepts of the problem domain and its solution;

(b) the means used to represent and reason about these concepts, namely:

- a structured set of attributes used to describe the basic concepts;
- a network of inter-related rules in terms of these concepts;

(c) the means used to control the uncertainties in the model, namely Support Logic Programming.

3.2 FRIL

FRIL: A Support Logic Programming System is a powerful computer language for Artificial Intelligence applications. FRIL has been developed and marketed by FRIL Systems Limited based on research in Support Logic reasoning by the Information Technology Research Centre, University of Bristol.

FRIL is the only logic programming language with an in-built facility for dealing with uncertainty. It has all the features of PROLOG with the added advantages that it can handle knowledge bases containing uncertainties of a probabilistic, fuzzy and evidential nature. It can represent and reason with uncertain and incomplete knowledge.

Support Logic Programming is based on probability theory - but generalised to allow reasoning with probability intervals. It also includes representations for fuzzy sets and partially conflicting knowledge statements so that reasoning with vague and inexact statements and reconciling knowledge sources from multiple viewpoints can be effectively modelled.

[1] gives an introduction to the mathematics of Support Logic Programming. Detailed descriptions are to be found in [2 - 9].

Due to the diversity of uncertainty that prevails in the iMAPS problem domain it is considered important to retain as much generality and flexibility as possible in the means used to represent and reason with uncertain knowledge in the iMAPS conceptual model.

It is the author's view that, wherever possible, there should be freedom to model using the uncertainty paradigm most appropriate to the aspect of the problem under consideration.

Support Logic Programming is a model for dealing with rule-based systems involving probabilistic and fuzzy uncertainty using probability theory. It is a generalisation of probability and logic programming which makes approximations using intervals.

Support Logic Programming was chosen as the principal uncertainty paradigm to be used in the research since:

(a) the nature of the iMAPS domain knowledge is rule-based and uncertain;

(b) is the most general, flexible and expressive of the available options.

Support Logic Programming allows us to model using the uncertainty paradigm we feel is the most appropriate. It is ideally suited to deal with description and inference of issues relating to software.

This choice strongly suggests that the language FRIL be used as the implementation language for the iMAPS knowledge-based system.

3.3 Strategy

The iMAPS conceptual model is to be composed of a small number of linked (conceptual) sub-models. Each sub-model will address one of the chosen iMAPS foci. Breaking the model down in this way has the following advantages:

(a) the predictive power, see Section 1.5.2, of a sub-model specifically designed to deal with a single software attribute, namely its focus, is likely to be greater (in respect to

this attribute) than that of a general model which is not dedicated to a single attribute;

(b) the reduced scope of a sub-model makes modelling and checking more tractable;

(c) sub-models can be developed as separate activities, conducted either by different people or at different times.

The primary disadvantage of this approach is that it introduces the need to consider how best to integrate the sub-models to form the composite model.

3.3.1 Progressive enhancement

Each conceptual sub-model is to be developed and improved using a method of **progressive enhancement**. The method is depicted in Figure 3. It has four key aspects:

(a) the development of an initial version of a **conceptual sub-model**;

(b) the **implementation** of this sub-model as a prototype knowledge-based system using the facilities of FRIL for dealing with uncertain knowledge-structures and evidence;

(c) the **assessment** of the sub-model by the use of the knowledge-based system in:

- peer reviews;
- experiments involving data from systems;

(d) the **refinement** of the sub-model to take account of feedback from assessments.

The method has the following advantages:

(a) the techniques used in the conceptual model for the treatment of uncertainty enable imperfect knowledge of the problem domain to be used. This allows preliminary versions of the model and the KBS tool to be put forward before the problem domain is fully understood;

(b) the conceptual model can be demonstrated rather than merely described. This helps a much wider audience, than would otherwise be possible, to appreciate and understand the model's purpose and value;

(c) early implementation of the conceptual model as a KBS means that it can be checked both by review **and test** at a very early stage;

(d) feedback from assessments can be used to improve understanding of the problem domain as well as to enhance the model and it's implementation.

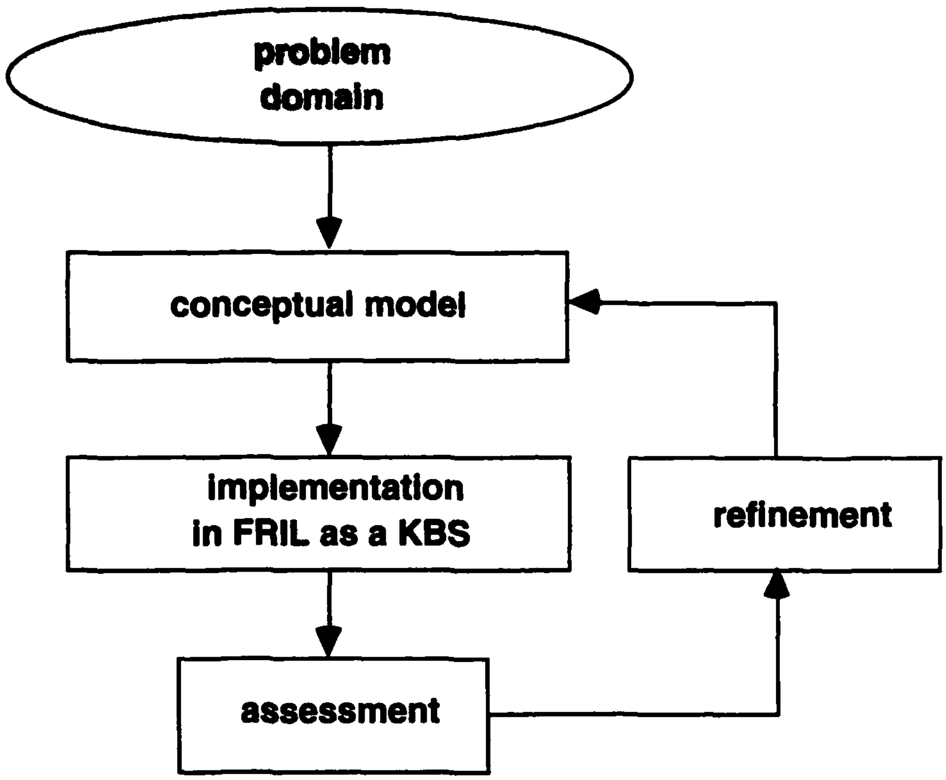

Figure 3 Development strategy

3.3.2 Integration

Linking or integrating the conceptual sub-models to form the composite conceptual model is an activity demanding considerable thought. Care must be taken to:

(a) avoid inconsistencies;

(b) avoid unnecessary replication in modelling and evidence collection;

(c) capitalise on the thinking and experience gained on preceding sub-models;

(d) ensure that incremental integration of the sub-models is performed in the most effective way.

Even when used in isolation from one another the sub-models will not be entirely independent. Dependence will arise from commonalities in:

(a) content;

(b) structure;

(c) development methods;

(d) checking methods.

3.4 iMAPS conceptual model

The iMAPS conceptual model will provide a basis for:

(a) defining the software, or some part or aspect of the software, being investigated;

(b) describing the intended properties of the software being investigated;

(c) observing the realised properties of the software being investigated;

(d) comparing the intended and realised properties of the software being investigated;

(e) comparing the properties of the software being investigated with other software;

(f) inferring the values of the high-level attributes of the software being investigated given evidence regarding the values of its low-level attributes ;

(g) feeding back concerning the dependability of the iMAPS conceptual model itself.

It does this by providing:

(a) an abstract description of software in terms of a set of inter-related attributes;

(b) a set of rules which are used to infer indirect measurements from direct measurement information;

(c) a means for dealing with the uncertainty in:

- the knowledge of the relationships by which such inferences can be made;
- the evidence available for the particular software being assessed.

Each sub-domain of the iMAPS conceptual model will comprise:

(a) an attribute map;

(b) a rule map;

(c) evidence collection schemata.

3.4.1 Attribute maps

A software attribute is a feature or property of a software entity of interest.

An attribute map is a collection of software attributes organised to emphasis their inter-relationships. Each map has a single focus which is the most abstract attribute that it represents. Attribute maps tend to be hierarchical in structure.

Each attribute in the attribute map has the following characteristics:

(a) a linguistic label which summarises its meaning;

(b) a definition of its meaning;

(c) a numeric label which identifies its position in the map;

(d) a decomposition into sub-attributes.

An attribute map is a descriptive structure. It is a means of describing abstract attributes of an item of software in terms of more concrete, and therefore more readily measurable, attributes defined by the map. It provides a partial description of software which is robust/optimal for subsequent inferences regarding the particular software attribute under investigation. Other partial descriptions are possible and these may be preferred for inferences regarding other software attributes.

An attribute map is a directed graph. Attribute map nodes represent software attributes. Attribute map arcs represent the relationships between software attributes. A map may have more than one type of arc. The most common type of arc represents the relationship that one attribute is a sub-attribute of another.

3.4.2 Rule maps

In the Support Logic Programming paradigm knowledge is represented as a set of **facts and rules** organised in a knowledge-base. Facts and rules may be expressed in terms of crisp or fuzzy set variable instantiations.

Facts and rules are both instances of clauses. A clause is a list of one or more atoms. An atom is a list whose first element is a predicate symbol or relation name and whose remaining elements are terms. A term is a number, a constant, a variable or a list.

A fact is a clause which is not conditioned on any other clause in the knowledge-base. A fact is a simple clause comprising just one atom.

A rule is a clause which is conditioned on other clauses in the knowledge-base. A rule is a compound clause comprising more than one atom. The first atom in the clause is called the 'head' of the clause and the remaining elements the 'body' of the clause. In a rule clause the rule-head and rule-body respectively represent the consequent and antecedent of the rule.

In the Support Logic Programming paradigm a model of a domain, such as that of iMAPS, is constructed as set of facts and rules which express our knowledge of that domain.

A rule-map is a collection of rules which describe the sophisticated relationships which hold between the attributes in an associated attribute map. Since the state of knowledge of these relationships is imperfect, the rules reflect the uncertainty that this introduces.

A rule-map is an inference structure. It provides rules for the prediction of the values of high-level attributes in an associated attribute map given evidence regarding low-level attributes in that attribute map. In this sense a rule-map can be regarded as defining indirect measures for the (high-level) attributes in an associated attribute map.

The rules in the model are in terms of the actual and intended values of the attributes used to describe software. In practice, information concerning the attributes of the software being assessed may be lacking or imprecise. The rules also deal with the uncertainty that this introduces.

A rule-map is a directed graph. Rule-map nodes are predicates couched in terms of the attributes of an associated attribute map. Rule-map arcs represent logical relationships between these predicates. There are many different types of arc corresponding to the many types of logical relationships which are possible, eg., equivalence, implication, conjunction, etc., and each arc type is further characterised to reflect the strength with which the relationship holds. Because the rules are expressed in terms of the attributes there is a tendency for the 'shape' of a rule-map to mirror the 'shape' of its associated attribute map to a limited extent.

Rule-maps are depicted diagrammatically and are expressed in detail in a notational form using the rule-schema technique which was developed by the author in earlier research, [13].

The rule-map diagrams have the following characteristics:

(a) easily scanned;

(b) give an immediate feeling of the scale of the knowledge encapsulated in the rule-map;

(c) give a sense of bearings within the rule-map;

(d) readily understandable by reviewers.

A rule-schema is a specification of a rule. It comprises:

(a) an informal, natural language description of the rule written in **bold typeface**

(b) a formal statement of the rule expressed in the schema notation written in plain typeface.

In all cases the natural language description is an informal interpretation of the formal rule **and not vice versa.** The natural language description is provided for the convenience of the reader wishing to read the rule specification quickly. The formal statement defines the intended meaning of the rule precisely and unambiguously.

The rule-schema technique enables the precise and unambiguous specification of rules in terms of the constructs of Support Logic Programming. It has the advantages of being:

(a) clear and succinct;

(b) readily enhanced;

(c) readily implemented as FRIL code.

3.4.3 Evidence collection schemata

The model provides a means of handling evidence (quantitative or qualitative) about the software being investigated. Evidence must be given to the model by the user. The evidence is required to be in terms of the software attributes defined in the attribute map.

For any particular attribute the user may supply evidence regarding whether the software is as intended in terms of that attribute.

For some attributes the user may supply evidence as the values, or instances, of the actual and intended attributes of the software being assessed. The conceptual model has procedures which take these values as inputs to evaluate the extent to which the software being assessed is as intended in terms of this attribute.

Obtaining information about certain attributes may require the expert, subjective knowledge of experienced software engineers. Other information, such as that derived from testing, can be treated in a more objective manner.

Evidence collection schemata take account of the uncertainty in both the knowledge of the relationships by which such assessments can be made and the uncertainty in the evidence available for the particular software being assessed.

The user-interface:

(a) provides guidance on how to enter evidence;

(b) helps the user interpret the attribute definitions.

3.5 Prototypes

Each iMAPS conceptual sub-model will be implemented as a prototype knowledge-based system (KBS). Each KBS will comprise a kernel and a user-interface. The kernel will be written in the knowledge engineering language FRIL. The user-interface is unlikely to be written in FRIL but is more likely to be written using a standard product for generation of user-interfaces such as Motif.

The KBS kernels and their respective KBS user-interfaces will be developed separately with their integration being a distinct activity. All such KBS components will be developed using a process of progressive enhancement involving repeated checking and refinement.

3.5.1 Kernel

The kernel is the core of a knowledge-based system. The iMAPS kernels will be implemented in such a way as to ensure efficient performance and ease of maintenance. A side-effect of this is that, without their associated user-interfaces in place, users will not be able to interact with a iMAPS kernel unless they have detailed prior understanding of both the sub-model being implemented and the FRIL language.

3.5.2 User-interface

In the early stages of the iMAPS work, only basic functions will be implemented in the user-interfaces to the knowledge-based tools. As the need to promote the iMAPS approach outside the ITD research environment develops, and as people with diverse backgrounds are introduced to the use of the iMAPS tools, the need for greater levels of sophistication in their user-interfaces will increase. The research and development of user-interfaces which do justice to these requirements is an activity needing specialist skills and considerable effort. It would seem appropriate that a Sub-Task be set-up to address these issues.

3.5.2.1 Basic

The basic user-interfaces will need to be able to allow an engineer with no detailed experience of either the programming language or the conceptual model to use the tools in a straightforward manner. The basic user-interfaces will be required :

(a) to control all interaction with the kernel;

(b) to guide the user through an interactive consultation with the kernel;

(c) to help the user to provide evidence about the software being assessed,

(d) to perform limited validation of input data.

(e) to enable the user to request that the kernel makes an assessment or prediction of a attribute of interest at any stage in the consultation.

3.5.2.2 Sophisticated

More sophisticated versions of these user-interfaces may include:

(a) a means of representing the knowledge contained in the KBS kernel, and the reasoning processes that it performs, which allows the user to create a faithful 'mental model' of what the kernel does;

(b) a means for bi-lateral communication between the KBS kernel and the user which is in accord with this mental model;

(c) a help facility which provides basic information on how to use the tool;

(d) a 'guidance' facility which could suggest different ways of achieving given goals, give advice on which alternative is the best against prescribed criteria, etc.

3.5.3 Checking

As well as checking the implementation of the KBS kernels and their associated user-interfaces, prototype checking plays a significant role in the development of the underlying iMAPS conceptual model.

Reference [15] describes the strategy that will be adopted to check that:

(a) an iMAPS model is correctly implemented as the kernel of a knowledge-based system;

(b) the user-interface to a knowledge-based system kernel is adequately implemented;

(c) an iMAPS conceptual model adequately captures the nature of the problem and its solution by conducting experiments using the knowledge-based system.

There will be marked differences between the approach to checking kernels and the user-interfaces. The principal differences are:

(a) there will be more iteration cycles planned for the kernels than for the user-interfaces;

(b) the type of checking to be used on the kernels is more stringent and more formal than that to be used on the user-interfaces.

These differences reflect that:

(a) compared with its user-interface. a kernel is the relatively 'critical' feature of an iMAPS tool;

(b) a kernel is to be implemented to conform with a (comparatively) mature and formal specification;

(c) the user-interface is to be implemented by prototyping from an immature and informal specification.

There are seven types of checking planned to be used in the development of iMAPS KBS's. These are:

(a) informal unit testing;

(b) formal unit testing;

(c) scenario testing;

(d) branch testing;

(e) ad hoc testing;

(f) integration testing;

(g) acceptance testing.

For each checking type [15] gives a brief description of:

(a) what the check involves;

(b) the type of anomalies the check is intended to expose;

(c) typical action on the check exposing an anomaly.

3.5.4 Refinement

Multiple refinement cycles are planned. Major cycles will arise from each of the types of planned checking. Minor cycles can also be anticipated to account for:

(a) kernel/user-interface integration;

(b) performance tuning;

(c) unexpected problems/finding.

3.5.5 Configuration Management

Strict configuration management of the conceptual model, KBS kernel, KBS user-interface and all associated documentation will be essential. Measures will be taken to ensure that this is as easy to achieve in practice as is possible. These include:

(a) linking the KBS code and associated specification from the conceptual model;

(b) use of automatic configuration management tools where possible.

Although simple, practical means of ensuring basic configuration control should be able to be devised and applied relatively easily, it is anticipated that some difficult and subtle problems will be raised in this respect by work of this nature. It is considered appropriate that a separate Sub-Task be set up to address this issue.

3.6 Workshop

It is planned to organise and host a goal-oriented workshop at DSTO in a subject area of central relevance to the iMAPS Task. The goals of the workshop are to be specific to the iMAPS Task and its attendees are to be limited to those who are able and willing to contribute to achieving these goals.

It is the intention to invite a leading figure in the field to visit DSTO for 1-2 week period and in that time take a prominent role in the workshop. Funds are available for such a person to travel from overseas. Candidates who are being considered for this role are:

(a) Dr Bruce Pilsworth (University of Bristol and FRIL Systems Limited)

(b) Dr Nicholas Ashley (Brameur Limited);

(c) Dr Agnes Kaposi (Kaposi Associates);

(d) Dr Norman Fenton (City University);

(e) Dr Barbara Kitchenham (National Computing Centre).

Members of the Australian Software Measurement Community who are being considered as possible attendees include:

(a) Prof Ray Offen (Macquarie University);

(b) Prof Ross Jefferies (University of New South Wales);

(c) Dr June Verner (University of New South Wales).

4. Products

The major items to be produced by iMAPS are summarised below.

4.1 iMAPS

4.1.1 Papers

Reports and papers which will be produced include:

(a) task plan

(b) general introduction

(c) iMAPS Conceptual Model

(d) summary

(e) journal/conference paper

4.1.2 Software

Programs which will be produced include:

(a) iMAPS KBS Kernel

(b) iMAPS KBS User-Interface

4.1.3 Presentations

Topics which will be covered include:

(a) concepts of uncertainty;

(b) mathematics of uncertainty;

(c) uncertainty in software;

(d) mathematics of uncertainty in software;

(e) iMAPS concept;

(f) FRIL.

4.1.4 Workshop

A goal-oriented workshop on a subject area of central relevance to the iMAPS Task will be organised and hosted at DSTO.

4.2 Software Reliability

4.2.1 Papers

Reports and papers which will bc produced include:

(a) literature review paper

(b) tool assessments

(c) attribute map - modelling process

(d) attribute map - full description

(e) rule map - modelling process

(f) rule map - full description

(g) evidence schemata - modelling process

(h) evidence schemata - full description

(i) checking - strategy

(j) checking - results

(k) summary

(l) journal/conference paper(s)

4.2.2 Software

Programs which will be produced include:

(a) software reliability prediction KBS kernel

(b) software reliability prediction KBS user-interface

4.2.3 Presentations

Topics which will be covered include:

(a) software reliability prediction attribute map;

(b) software reliability prediction rule map.

4.3 Software Cost

4.3.1 Papers

Reports and papers which will be produced include:

(a) literature review paper

(b) tool assessments

(c) attribute map - modelling process

(d) attribute map - full description

(e) rule map - modelling process

(f) rule map - full description

(g) evidence schemata - modelling process

(h) evidence schemata - full description

(i) checking - strategy

(j) checking - results

(k) summary

(l) journal/conference paper(s)

4.3.2 Software

Programs which will be produced include:

(a) software cost estimation KBS kernel

(b) software cost estimation KBS user-interface

4.3.3 Presentations

Topics which will be covered include:

(a) software cost estimation attribute map;

(b) software cost estimation rule map.

5. Collaboration

iMAPS was to be one of the five research themes of a proposed CRC in Software and Systems Engineering. CRCSSE failed to gain backing from the authorities. Nevertheless, the effort invested in forging the partnership and preparing the proposals has been beneficial in establishing a common understanding regarding some of the important issues of iMAPS. The opportunity still exists to capitalise on this establishing a common understanding,

The most likely useful collaborators emerging from this source are:

(a) DSTO - SE Group, ITD (Landherr, Vernik, Burke);

(b) Macquarie University - JRCASE (Offen);

(c) University of New South Wales (Jefferies, Verner).

The less likely collaborators emerging from CRCSSE are:

(a) CSIRO - IISE (Frater, O'Sullivan);

(b) Flinders University - JRCIT (Marlin, Mudge).

Contact with the following organisations from overseas will also be maintained:

(a) METKIT (Ashley, Kaposi, Fenton);

(b) City University, London - Centre for Software Reliability (Fenton, Littlewood et al)

(c) QUINCE (Hufton, Lockett, Pilsworth);

(d) MERMAID (Kitchenham).

Liaison with the following groups is also expected:

(a) University of South Australia, ICSEA (Sobey);

(b) CSIRO (Veevers).

6. Commercialisation

iMAPS does not have any immediate commercial potential although it does present several possibilities for income generation in the future through the supply of products and the provision of associated services on a commercial basis. The best strategy to exploit these possibilities is not clear at present. Careful consideration of this will be made as the Task progresses and proposals will be made as appropriate.

Intellectual property produced by the Task will need to be protected, although publication of basic ideas and results is also desirable. All software developed will be copyright and, where appropriate, the details of the conceptual model and source code will be treated as a trade secret. Advice will be sought at an early stage in the Task regarding the best way of achieving these goals.

7. Discussion

iMAPS is a goal-driven, applied science research project which sets out to devise an integrated approach to the description, measurement, assessment and prediction of the attributes of software.

iMAPS is therefore highly relevant to Defence projects which involve software as well as being of general interest to the software engineering community worldwide.

Software is an increasingly important element in Defence systems. If iMAPS is successful it will benefit Defence systems projects, both now and in the future, by enabling greater control, better risk management, improved quality and reduced costs. The value of these benefits is to be expected to increase with time.

The scope of iMAPS is enormous and its subject matter is both demanding and intricate. iMAPS work will involve:

(a) fundamental scientific research into software description, measurement, assessment and prediction;

(b) knowledge engineering research concerned with the development of tools and techniques;

(c) application of the results of this research to Defence problems.

Some aspects of iMAPS have been tackled before with varying degrees of success. For example, software measurement is becoming an area of prominence in software engineering research now that underlying theoretical foundations have been put in place, [17, 21]. Other aspects have hardly been investigated at all. For example, software description is a topic which is almost entirely unexplored at present. The two characteristics of iMAPS which distinguish it from other work are:

(a) it aims to be an integrated approach;

(b) it aims to treat uncertainty in a rigorous manner.

For iMAPS to succeed it is essential that it be carefully planned and risk managed. Some aspects of the work are relatively low risk but others are very ambitious and speculative. The Task promises short, medium and long term payback. The short term benefits are associated with relatively low risks, thus maximising the likelihood of the Task making a favourable impact quickly and affording the opportunity for redirection of the work if preliminary results indicate that this is necessary. The anticipated benefits are discussed in following sections.

7.1 Short term benefits

The benefits expected to be accrued within the first two years of the Task as currently planned are as follows.

7.1.1 Support capability

ITD's capability to support Defence in the application of the iMAPS related technologies developed externally will be greatly enhanced as a result of the investigations planned for the early phases of the Task. The surveys and assessments of software reliability modelling and software cost estimation methods and tools will be contribute significantly in this respect.

7.1.2 Communication

The iMAPS conceptual model will provide a framework for:

(a) describing software in a precise and unambiguous way;

(b) measuring software;

(c) reasoning about software in a consistent, repeatable and auditable manner;

(d) quantifying the uncertainties that prevail in the procurement, development and use of software.

It therefore has the potential to provide a medium for:

(a) the clear exchange of many types of thought regarding software;

(b) the conduct of rational argument about software.

This framework has immediate application in support of discussions and investigations of both specific software systems, and software in the general sense. By providing software engineers with a language in which the problems, concerns and ideas may be clearly and unambiguously expressed, the iMAPS conceptual model can contribute immediately to improving software products.

7.1.3 Quality assurance and quality control

The iMAPS conceptual model allows evidence (or lack of evidence) regarding how software has been made and checked to be assessed in context. This capability can be used to:

(a) assess whether software systems have been sufficiently checked for their intended purpose;

(b) assess the coverage of quality control and quality assurance activities to establish:

- the adequacy of the software for its purpose;
- areas requiring further checking.

7.1.4 Metrics programme

Modern software engineering development uses metrics programmes to gather, collate and analyse data regarding their products. Software metrics programmes have been shown to contribute towards improvements in productivity, quality control, estimating etc. It is suggested that iMAPS will find immediate application in setting up and running a software metrics programme since:

(a) the conceptual model provides a framework for collecting and collating software metrics;

(b) the prototype tool enables conclusions to be drawn using the metrics data.

Information gained during the course of such a programme would provide valuable feedback to be used to refine the conceptual model and tool.

7.2 Medium term benefits

Assuming that development of both the conceptual model and the assessment tool continue beyond the planned duration of the current Task, and that the model is put to the uses outlined above, then the state of the work in the medium term (say 2 to 4 years) is postulated to be as follows:

(a) the conceptual model will have been refined through various reviews and enhancements, it will allow the use of any, or all, evidence which may generally be anticipated to be available, in a form appropriate to the evidence concerned (ie without the need for a user to interpret the evidence for the model);

(b) the assessment tool will have been refined through various checks and enhancements, in particular, the user interface will provide a robust, user friendly means of conducting sophisticated consultations with the KBS kernel;

(c) the model will have been given exposure to industrial and academic fora. The seeds of the model's concepts and associated terminology will have been sown. Feedback from such exposure will have been used to enhance the model. Software engineers working locally to where the model is being developed will begin to use its ideas and terminology as well as advising of practical considerations;

(d) experience of iMAPS assessments of attributes, such as software dependability, conducted on various systems will be beginning to lead to an awareness of "acceptable" and "unacceptable" values of these attributes.

As well as the uses identified above, it is anticipated that at this interim stage in the development of the iMAPS conceptual model the following applications would be appropriate.

7.2.1 Quality assurance and quality control

The iMAPS approach could be applied routinely and formally within a quality programme to ensure that software products are adequately checked before they are released or used.

The iMAPS tools would support decisions regarding whether:

(a) software products are acceptable or unacceptable for their intended purpose;

(b) further checking is required.

7.2.2 Decision driver

iMAPS assessments may be carried out by, or on behalf of, software engineers or managers. Such "what if" assessments may be conducted as a means of predicting the impact of design or managerial decisions on the properties of software products. This will help to guide engineers and managers towards decisions which achieve the best values of these properties for a given set of constraints.

7.3 Long term benefits

If the iMAPS work continues, and the uses discussed above take place, then it is anticipated that in the long term (say 10 years) that:

(a) the iMAPS conceptual model will be mature and will capture the understanding of a number of experts;

(b) mature iMAPS tools will be available and receiving extensive use. It is possible that these tools may be part of an integrated project support environment;

(c) the concepts and terminology of the iMAPS conceptual model will be widely understood and accepted within the software engineering community. A discipline of iMAPS engineering may be emerging;

(d) the significance of particular values of the iMAPS attributes eg dependability will be generally appreciated.

There are various uses which can be envisaged for the iMAPS model and tool at this stage.

7.3.1 Expression of requirements

Procuring authorities may make statements of requirement for software systems which specify:

(a) the required values of various key attributes of the software;

(b) the processes used to make and check the software;

(c) the required coverage of quality control and quality assurance activities.

The iMAPS approach will make it possible to make **testable** statements of requirements which at the present time can only be made in loose terms. Furthermore, the assessment model will provide a mechanism whereby achievement of the requirements may be demonstrated for acceptance purposes.

7.3.2 Design Driver

At such time as software requirement specifications are made in terms of quantified and testable statements regarding the values of software attributes, software producers will be forced to use an iMAPS approach when making management and design decisions. The iMAPS tool will:

(a) support this decision process;

(b) promote an understanding of how to plan to produce software with designated properties within project constraints.

A sophisticated user interface may be able to prompt the user to provide additional evidence regarding particular attributes in order to reduce uncertainty, or highlight particular attributes as being the source of significant adverse effects.

7.4 Anticipated problems

The success of every endeavour is threatened by problems being encountered which cannot be overcome. However, the risk of the endeavour failing can be reduced if the problems can be anticipated and dealt with before they become manifest. Clearly, it is rarely possible to anticipate all the problems that can occur in any given situation. The problems which have been recognised as being of significance to iMAPS are discussed below. The nature of their likely solutions is outlined wherever possible.

7.4.1 Misconceptions

It is possible that a misplaced expectations could form amongst iMAPS stakeholders that iMAPS will in itself reduce uncertainty in the software that it is used to investigate. This is not the case. iMAPS sets out to deal with uncertainty in software in a rigorous manner. If high levels of uncertainty exist in an analysis then these will be shown by iMAPS. However, iMAPS is intended to be used as a tool to direct effort in the control and reduction of uncertainty and in this way can contribute to the reduction of uncertainty in software.

It is possible that the iMAPS team may experience difficulties and reluctance from many quarters when they try to develop understanding of, and acceptance and commitment to, the novel methods that iMAPS professes. For example, there is a widespread and misplaced confidence in the scientific and engineering community that uncertainty of all types can be described and handled using probability theory. Previous experience shows that even when these problems are explained, and solutions identified, that it is common for people to stick doggedly to the 'accepted' view - even though it has been demonstrated to be wrong.

7.4.2 Uncertainty modelling

There are various uncertainty modelling issues which will have to be approached with care in the development of the iMAPS conceptual model. Those of principal concern are introduced below.

7.4.2.1 Uncertainty explosion

Inferences drawn from rule-maps which model (highly) uncertain knowledge can exhibit the phenomenon of uncertainty explosion in which the propagation of uncertainty through a chain of reasoning occurs in a way that is dramatically greater than expected. This occurs most commonly when groups of inter-related rules, each of which models an item of knowledge in which there is relatively low uncertainty, transmit uncertainty in their inferences from one to another in such a manner that the uncertainty in the ultimate conclusion, that is at the end of the chain of inferences, is surprisingly large.

Uncertainty explosion does not necessarily occur as a consequence of incorrect modelling but more usually as a consequence of a rule-map being too sparse. Uncertainty explosion it is usually an undesirable effect but one which can often be avoided by thoughtful construction of the rule-map. Uncertainty explosion can be controlled or avoided by:

(a) using knowledge which is free from uncertainty;

(b) building 'redundancy' into the model whereby diverse items of uncertain knowledge addressing the same issue are used in concert, with the aggregate effect that uncertainty in inferences drawn by using them together is significantly less then in any inference drawn from one of them in isolation;

(c) structuring the rule-map so that none of its components is too large or complex.

7.4.2.2 Checking uncertain predictions

There will be unavoidable, theoretical difficulties involved in checking the validity of some iMAPS predictions which are qualified by a measures of uncertainty. For example, consider an estimate of software development costs given as:

$$P(\$m\ 2.8 \leq costs \leq \$m\ 3.4) = 0.95$$

That is, there is a 95% probability that the development costs lie between \$m 2.8 and \$m 3.4. Traditional, objective statistical techniques are inappropriate in checking this prediction as there is no opportunity to repeat the development project under identical initial conditions and thus no possibility of constructing a sample of actual costs on which to base a test of the correctness of the estimate.

Subjective assessment of the estimates will often be the only approach which can be used. This may involve:

(a) examination of the assumptions, data and methods used to make the estimate;

(b) testing the predictive qualities of the techniques being used on idealised cases or especially designed experiments.

7.4.3 Implementation

7.4.3.1 FRIL

The FRIL language was developed by a small, university based group who subsequently set up a company to market and support their product. Problems may arise which include:

(a) the FRIL product is unlikely to be as mature and stable as is ideal for a development project;

(b) the quality and timeliness of support is unlikely to be the same as from mainstream commercial organisations. This may not prove to be a problem but, on the contrary, prove to be to iMAPS' advantage;

(c) FRIL Systems Limited may not be a stable enterprise and if it were to cease to exist then the iMAPS work would be jeopardised.

7.4.3.2 Rate of evolution

It is possible that there could be a considerable mismatch between the natural rate of evolution of the iMAPS conceptual model and the rate at which the KBS (both kernel and user interface) can be enhanced. Various difficulties could arise if this situation is not addressed and a workable solution found. These include:

(a) loss of configuration control;

(b) unacceptably retarded rate of refinement of the conceptual model.

The Configuration Management Sub-Task will consider these issues.

7.4.3.3 User-interface sophistication

A very sophisticated user-interface will be required to do full justice to the subtlety of the conceptual model/KBS kernel. Development of ideas and techniques to produce an adequate user-interface and the subsequent implementation of these ideas as a program will require both insight and effort. Failure to achieve this is likely to result in the iMAPS tools giving false impressions of the value of the iMAPS approach.

The User Interface Sub-Task will consider these issues.

7.4.4 Resources

Even if the iMAPS Task is carefully scoped and managed then its effort requirements will be large. Involvement of other parts of Defence will be needed after the preliminary phases if the Task is to make headway at an adequate rate.

THIS IS A BLANK PAGE

References

1. M.M. Burke, 'Uncertainty Analysis', Information Technology Divisional Paper ITD-93-12, September 1993.

2. J.F. Baldwin, B.W. Pilsworth and T.P. Martin, 1988, 'Support Logic Programming' in FRIL manual, Chapter 6.

3. J.F. Baldwin, 'Support Logic Programming', in Fuzzy Sets Theory and Applications, A. Jones et al (eds.), pp 133-170, D. Reidel Pub. Co.,1986, (NATO ASI Series, Series C: Mathematical and Physical Sciences Vol. 177).

4. J.F. Baldwin, 'Evidential Support Logic Programming', Fuzzy Sets and Systems 24, 1987, pp 1-26, Elsevier Science Publishers, North-Holland.

5. J.F. Baldwin, ITRC 67, 'Support Logic Programming for Expert Systems'.

6. J.F. Baldwin, ITRC 76, 'An Uncertainty Calculus for Expert Systems'.

7. J.F. Baldwin, ITRC 93, 'Expert Systems and Evidential Reasoning'.

8. J.F. Baldwin, ITRC 109, 'Evidence Theory, Fuzzy Logic and Logic Programming'.

9. J.F. Baldwin, 'FRIL - An Inference Language based on Fuzzy Logic', University of Bristol Internal report EM/FS44.

10. AdaMAT/D Reference Manual, Dynamics Research Corporation, Systems Division, 60 Frontage Road, Andover, Massachusetts 01810.

11. L.N. Harris, 'The use of models in reliability', in 'Software Reliability - achievement and assessment', edited by B. Littlewood, Blackwell Scientific Publications, 1987.

12. R.J. Vernik and M.M. Burke, 'Perspective-Oriented Description: Integrating and Tailoring Information for Software Engineering', Procs. 6th Conf. on Software Engineering and its Applications, Paris, France, November, 1993.

13. M.M. Burke, 'Software Dependability Assessment', PhD Thesis, Department of Engineering Mathematics, University of Bristol, UK, 1991.

14. R.J. Vernik and M.M. Burke, 'A conceptual model for software engineering description', in preparation.

15. M.M. Burke, 'Checking knowledge-based systems involving uncertainty', in 'Dependability of Artificial Intelligence Systems (DAISY-91)', G.H. Schildt and J Retti (Editors), Elsevier Science Publishers B.V. (North-Holland), 1991.

16. M.M. Burke, 'iMAPS Task Plan', DST 93/949, July 1993.

17. N.E. Fenton, 'Software Metrics: A Rigorous Approach', Chapman & Hall, 1991.

18. J.C. Laprie, 'Dependable Computing and Fault-Tolerant Systems Vol 5., Dependability: Basic Concepts and Terminology', Springer-Verlag, 1992.

19. R.J. Vernik and I. Turner, 'Techniques and Tools for Analysing Software Products', The Australian Computer Journal, Vol 24, No. 3, August, 1992.

20. B.W. Boehm, 'Software Engineering Economics', Prentice-Hall, New York, 1981.

21. A. Kaposi and I. Pyle, 'Systems are not only software', Software Engineering Journal, January, 1991.

22. P.H.M. Kok and B.A. Kitchenham, 'Enhanced support for cost estimation of software development', Journal of Software Research, Volume 2, No 1, pp 32 - 37, 1990.

23. P.H.M. Kok, 'New approach to software cost estimation', Journal of Software Research, Volume 2, No 3, pp 78 85, 1990.

24. R.J. Vernik and S.F. Landherr, 'Lessons Leraned from Quality Evaluations of Nulka Software', ERL-0761-RE, October 1993.

25. R.J. Vernik, 'Software Measurement and Analysis Technology Task Plan', ALO 91/534, October 1993.

26. A.A. Kaposi, 'Measurement theory', in 'Software Engineer's Reference Book', edited by J.A. McDermid, Butterworth-Heinemann, 1991.

27. R.J. Vernik, I. Turner, C. Baker and S.F. Landherr. 'Automated Support for Assessment of Large Ada Software Systems', in Proceedings of TRI-ADA'91 Conference, 1991.

28. R.J. Vernik and I. Turner, 'Software Reuse: Can It Deliver?' in Proceedings of 7th Australian Conference on Software Engineering (ASWEC'93), Sydney, Australia, October 1993.

29. 'SEE-Ada User's Guide, Version 2.3', A21-718-0497-001, January 1994.

30. M. Bush and N. Ashley, 'METKIT: Toolkit for Metrics Education', IEEE Software, November 1993.

31. IIT Research Institute, 'A descriptive evaluation of software sizing models, Data Analysis Centre for Software', RADC/COED, Griffiths AFB, NY 13441, USA, 1987.

32. K.H.L. Ho, B.W. Pilsworth and M.M. Burke, 'A Knowledge Engineering Approach for Software Cost Estimation', University of Bristol ITRC Report, April 1991.

33. M.M. Burke, 'A sparse conceptual model for software development cost estimation', DTI Project QUINCE Report, QUINCE/W/20, February 1991.

34. A.D. Carlton et al, 'Software Measurement for DoD Systems: Recommendations for Initial Core Measures', CMU/SEI-92-TR-19, ESC-TR-92-019, September 1992.

35. IEEE Std 1061-1992, 'IEEE Standard for a Software Metrics Methodology', 3 December 1992.

36. IEEE Std 982.1-1988. 'Standard Dictionary of Measures to Produce Reliable Software, 9 June 1988.

37. IEEE Std 982.2-1988. 'Guide for Use of Standard Dictionary of Measures to Produce Reliable Software (IEEE Std 982.1-1988)', 27 September 1988.

38. US Army Communications -Electronics Command, Software Engineering Directorate, 'Streamlined Integrated Software Metrics Approach (SISMA) Guidebook', 12 July 1993.

40. AMI, 'AMI Handbook: A Quantitative Approach to Software Management', ISBN 0 9522262 0 0, 1992.

THIS IS A BLANK PAGE

DISTRIBUTION

	No. of Copies
Defence Science and Technology Organisation	
Chief Defence Scientist)	
Central Office Executive)	1 shared copy
Counsellor, Defence Science, London	Cont Sht
Counsellor, Defence Science, Washington	Cont Sht
Senior Defence Scientific Adviser	1 copy
Scientific Adviser POLCOM	1 copy
Director, Aeronautical & Maritime Research Laboratory	1 copy
Navy Office	
Naval Scientific Adviser	1 copy
Army Office	
Scientific Adviser, Army	1 copy
Airforce Office	
Air Force Scientific Adviser	1 copy
Defence Intelligence Organisation	
Assistant Secretary Scientific Analysis	1 copy
Electronics & Surveillance Research Laboratory	
Director	1 copy
Chief Information Technology Division	1 copy
Chief Electronic Warfare Division	Cont Sht
Chief Guided Weapons Division	Cont Sht
Chief Communications Division	Cont Sht
Chief Land, Space and Optoelectronics Division	Cont Sht
Chief High Frequency Radar Division	Cont Sht
Chief Microwave Radar Division	Cont Sht
Chief Air Operations Division	Cont Sht
Chief Maritime Operations Division	Cont Sht
Research Leader Command & Control and Intelligence Systems	1 copy
Research Leader Military Computing Systems	1 copy
Research Leader Command, Control and Communications	1 copy
Manager Human Computer Interaction Laboratory	Cont Sht
Head, Program and Executive Support	Cont Sht
Head, Software Engineering Group	2
Head, Trusted Computer Systems Group	Cont Sht
Head, Command Support Systems Group	1
Head, Intelligence Systems Group	Cont Sht
Head, Systems Simulation and Assessment Group	Cont Sht
Head, Exercise Analysis Group	Cont Sht

Head, C3I Systems Engineering Group	Cont Sht
Head, Computer Systems Architecture Group	Cont Sht
Head, Information Management Group	Cont Sht
Head, Information Acquisition & Processing Group	Cont Sht
Author	1 copy
Publications & Publicity Officer ITD	1 copy

Libraries and Information Services

Australian Government Publishing Service	1 copy
Defence Central Library, Technical Reports Centre	1 copy
Manager, Document Exchange Centre, (for retention)	1 copy
National Technical Information Service, United States	2 copies
Defence Research Information Centre, United Kingdom	2 copies
Director Scientific Information Services, Canada	1 copy
Ministry of Defence, New Zealand	1 copy
National Library of Australia	1 copy
Defence Science and Technology Organisation Salisbury, Research Library	2 copies
Library Defence Signals Directorate Canberra	1 copy
British Library Document Supply Centre	1 copy
Parliamentary Library of South Australia	1 copy
The State Library of South Australia	1 copy

Spares

Defence Science and Technology Organisation Salisbury, Research Library	6 copies

Department of Defence

DOCUMENT CONTROL DATA SHEET

1. Page Classification	Unclassified
2. Privacy Marking/Caveat (of document)	

3a. AR Number AR-008-867	3b. Laboratory Number ERL-0826-GD	3c. Type of Report General Document	4. Task Number DST 93/208
5. Document Date AUG 1994	6. Cost Code	7. Security Classification	8. No. of Pages: 60 9. No. of Refs.: 40

10. Title

iMAPS GENERAL INTRODUCTION

7. Security Classification

* U	U	U
Document	Title	Abstract

S (Secret) C (Confi) R (Rest) U (Unclass)

* For UNCLASSIFIED docs with a secondary distribution LIMITATION, use (L) in document box.

11. Author(s)

Martin Burke

12. Downgrading/Delimiting Instructions

13a. Corporate Author and Address

Electronics & Surveillance Research Laboratory
PO Box 1500, Salisbury SA 5108

13b. Task Sponsor

DSTO

14. Officer/Position responsible for

Security:..

Downgrading:..

Approval forRelease:...DERL..

15. Secondary Release Statement of this Document

APPROVED FOR PUBLIC RELEASE

16a. *Deliberate Announcement*

No Limitation

16b. Casual Announcement (for citation in other documents)

[X] No Limitation [ ] Ref. by Author , Doc No. and date only.

17. DEFTEST Descriptors

*IMAPS
Software engineering
Computer program reliability
Performance evaluation

18. DISCAT Subject Codes

12

19. Abstract

This paper introduces the main ideas of the iMAPS Task which aims to establish an integrated approach to the description, measurement, assessment and prediction of the attributes of software.